Vnořené dotazy

Vnořené dotazy

- Vnořeným dotazem (poddotazem) je SELECT, který je součástí jiného SQL příkazu. Výsledek vnořeného dotazu je použit nadřazeným dotazem.
- Kde lze použít vnořený dotaz?
  - Jako dočasně vytvořenou tabulku v klauzuli FROM SELECTu,
  - kdekoliv je očekáván výraz, pokud vnořený SELECT vždy vrací jednu hodnotu,
  - za operátory IN, ANY (SOME), ALL, EXISTS
  - v Oracle prakticky „všude“, i v seznamu sloupců, které se vybírají SELECTem.
- Vnořený SELECT je potřeba uzavřít do kulatých závorek.

Použití vnořeného dotazu

- Vnořený dotaz použijeme:
  - V případě složitějších dotazů, kdy potřebujeme nejprve získat určitou informaci (předzpracovat data), která se nám hodí pro použití v dalším dotazu. Z hlediska výkonu je ale vždy výhodnější zkonstruovat jeden větší dotaz než více postupných dotazů!
  - Pokud pro každý zpracovávaný záznam musíme zjistit dodatečnou informaci, kterou nejde jednoduše získat přímo v nadřazeném dotazu.
- Př.:

```
SELECT jmeno, prijmeni, plat FROM zamestnanec
WHERE plat > (SELECT AVG(plat) FROM zamestnanec);
```

Použití vnořeného dotazu

- Příklad vnořeného dotazu, který vrací více hodnot:

```
SELECT prijmeni, jmeno, zamestnani_id
 FROM zamestnanec
 WHERE zamestnani_id IN (
    SELECT zamestnani_id FROM zamestnanec
    WHERE jmeno = 'Karel');
```

- Pokud dotaz vrací více hodnot, nesmí být uveden na místě, kde je očekávána jen jedna hodnota. Dotaz by skončil chybou, nebo výsledky by byly nepředvídatelné. Vnořený dotaz vracející více hodnot umí zpracovat operátory IN, ANY, ALL, EXISTS atd.
- Když se vnořuje více vnořených dotazů, tak nejvnitřnější se vykonává jako první.

Použití vnořeného dotazu

- Příklad vnořeného dotazu, který je vyhodnocován pro každý řádek nadřazeného dotazu:

```
-- Výběr zaměstnanců, kteří mají ve
-- svém oddělení podprůměrný plat
SELECT prijmeni, jmeno, plat
FROM zamestnanec z_nadrazeny WHERE plat < (
  SELECT AVG(plat)
  FROM zamestnanec z_vnoreny
  WHERE z_vnoreny.zamestnani_id
        = z_nadrazeny.zamestnani_id);
```

- V některých případech je možné takto vyhodnocovaný vnořený dotaz nahradit spojováním tabulek, které je v databázových systémech vysoce optimalizované a zpravidla výkonnější.

Použití vnořeného dotazu

- Pokud vrací vnořený dotaz pouze jednu hodnotu (skalární poddotaz), lze použít i přímo ve vracených sloupcích dotazu.
- Příklad použití skalárního poddotazu:

```
SELECT prijmeni, jmeno,
   (SELECT AVG(plat) FROM zamestnanec) AS prumerny_plat
FROM zamestnanec;
```

Klíčové slovo WITH

- U složitějších dotazů je skládání dotazu jako na předcházejících snímcích nepřehledné. Také co když výsledek poddotazu potřebujete použít vícekrát? Z toho důvodu vzniklo klíčové slovo WITH:

SELECT začíná klíčovým slovem WITH

Název pomocné tabulky

```
WITH platy AS
  (SELECT AVG(plat) prumer FROM zamestnanec
  )
SELECT jmeno,
  prijmeni,
  plat
FROM zamestnanec
CROSS JOIN platy
WHERE plat > prumer
```

Začátek skutečného SELECTu

Pomocná tabulka platy má jeden sloupec s názvem prumer. Je v něm jeden řádek s hodnotou průměrného platu všech zaměstnanců

S tabulkou platy se pracuje jako s každou jinou tabulkou. Zde je jedno z mála rozumných použití kartézského součinu

Všechny cesty vedou do Říma I.

```
-- cost = 5
select jmeno, prijmeni
from zamestnanec
left join pilot
using (zamestnanec_id)
where pilot_id is null;
```

```
-- cost = 6
select jmeno, prijmeni
from zamestnanec
minus
select jmeno, prijmeni
from zamestnanec
join pilot
using (zamestnanec_id);
```

Na těchto dvou stránkách jsou různé způsoby řešení příkladu:
Vypište všechny zaměstnance, kteří nejsou piloti.

Všechny cesty vedou do Říma II.

```
-- cost = 2
select jmeno, prijmeni
from zamestnanec
where zamestnanec_id
  not in (
    select zamestnanec_id
    from zamestnanec
    join pilot
    using (zamestnanec_id));
```

```
-- cost = 2
select jmeno, prijmeni
from zamestnanec z
where not exists (
    select 1 from pilot p
     where p.zamestnanec_id =
           z.zamestnanec_id
    );
```

VNOŘENÉ DOTAZY V DML / DDL

UPDATE

- Zvýšení platu o 10% všem zaměstnancům, kteří mají ve firmě podprůměrný plat:

```
UPDATE zamestnanec
SET plat   = ROUND((plat / 10)) + plat
WHERE plat < (SELECT AVG(plat) FROM zamestnanec)
```

DELETE

- Smazání všech destinací, kam nejsou naplánovány žádné lety:

```
DELETE FROM destinace a
WHERE a.destinace_id IN
  (SELECT b.destinace_id
  FROM destinace b
  LEFT JOIN zastavka
  ON zastavka.destinace_id  = b.destinace_id
  WHERE cislo_letove_linky IS NULL
  )
```

CREATE TABLE AS

- Na základě výsledku vnořeného SELECTu je možné vytvořit novou tabulku:

```
CREATE TABLE zamestnanci_archiv AS
  (SELECT * FROM zamestnanec
   WHERE datum_ukonceni IS NOT NULL)
```

- Do této nové tabulky jsou zkopírovány pouze omezení NOT NULL, ostatní omezení zkopírovány nejsou.

INSERT

- Je také možné použít vnořený SELECT při vkládání záznamů do tabulky:

```
INSERT INTO zamestnanci_archiv
(SELECT * FROM zamestnanec WHERE aktivni = 'n')
```

WITH CHECK OPTION

- Klíčové slovo WITH CHECK OPTION nepovolí změnit řádky, které nevyhovují podmínce v poddotazu:

```
INSERT
INTO
  ( SELECT jmeno, prijmeni, problematicky
    FROM pasazer
    WHERE problematicky = 'n'
    WITH CHECK OPTION
  )
  VALUES ('Jirka', 'Pinkas', 'y');
```

- Tento INSERT se neprovede úspěšně!

ROWNUM

- Pseudo-sloupec, který obsahuje pořadové číslo záznamu v dotazu.
- Čísluje se 1, 2 , 3 , … , N.
- Používá se pro stránkování záznamů (typické pro webové aplikace), pro zpracovávání omezeného množství záznamů (top-N processing) a pro ladění dotazů.
- ROWNUM je dostupný pouze v databázi Oracle.
  V některých jiných databazích lze nalézt klauzuli LIMIT s podobnou funkcionalitou.
- Příklad použití:

```
SELECT ROWNUM AS poradi, jmeno, prijmeni
FROM zamestnanec WHERE ROWNUM < 16;
```

ROWNUM

- Při použití pseudo-sloupce ROWNUM je potřeba brát v úvahu postup zpracování dotazu databází:
  - Jako první se provede FROM a WHERE klauzule,
  - proběhne přiřazení ROWNUM všem záznamům, které vyhovují FROM/WHERE klauzuli,
  - aplikuje se projekce (výběr atributů z klauzule SELECT),
  - aplikuje se GROUP BY,
  - aplikuje se HAVING,
  - aplikuje se ORDER BY.
- Při opomenutí postupu zpracování může dotaz vracet chybné záznamy.

Použití ROWNUM

- Chybné použití řazení v kombinaci s ROWNUM. Řazení je provedeno až po výběru prvních 10 záznamů, které jsou vybrány „náhodně“ (to většinou není to, co chceme):

```
SELECT jmeno, prijmeni FROM zamestnanec
WHERE ROWNUM <= 10
ORDER BY prijmeni, jmeno;
```

- Správné použití řazení (Top-N Reporting):

```
SELECT * FROM (
    SELECT jmeno, prijmeni FROM zamestnanec
    ORDER BY prijmeni, jmeno
)
WHERE ROWNUM <= 10;
```

- Dotaz, který nevrátí žádné záznamy (ROWNUM není v době provádění WHERE záznamům vůbec přiřazeno):

```
SELECT jmeno, prijmeni FROM zamestnanec
WHERE ROWNUM > 1;
```

Použití ROWNUM pro stránkování

```
SELECT *
FROM (
    SELECT z.*, ROWNUM AS poradi
    FROM (
        SELECT jmeno, prijmeni, plat
        FROM zamestnanec
        ORDER BY prijmeni, jmeno
    ) z
    WHERE ROWNUM <= 20
)
WHERE poradi > 10;
```

Poznámky I.

- Oracle 11g Optimizátor rozpozná top-n dotaz a netřídí všechny záznamy v poddotazu.

Poznámky II.

- V jiných databázích (MySQL, MSSQL, …) jsou obdobou ROWNUM klíčová slova LIMIT a OFFSET nebo TOP.
- Od Oracle 12c existuje FETCH FIRST 10 ROWS ONLY, což je také modernější varianta tvorby top-n processing:

```
SELECT eno,ename,sal FROM emp ORDER BY SAL DESC
FETCH FIRST 10 ROWS ONLY;

SELECT eno,ename,sal FROM emp ORDER BY SAL DESC
FETCH FIRST 10 PERCENT ROWS ONLY;

SELECT eno,ename,sal FROM emp ORDER BY SAL DESC
OFFSET 5 ROWS FETCH NEXT 5 ROWS ONLY;
```